SANDIA REPORT

SAND2004-6198
Unlimited Release
Printed January 2005

Automated Visual Direction LDRD 38623 Final Report

Robert J. Anderson

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94AL85000.

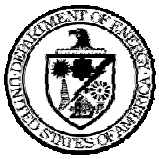
Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online



SAND2004-6198
Unlimited Release
Printed January 2005

Automated Visual Direction
LDRD 38623 Final Report

Robert J. Anderson
Mobile Robotics Department
Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-1125

Abstract

Mobile manipulator systems used by emergency response operators consist of an articulated robot arm, a remotely driven base, a collection of cameras, and a remote communications link. Typically the system is completely teleoperated, with the operator using live video feedback to monitor and assess the environment, plan task activities, and to conduct the operations via remote control input devices. The capabilities of these systems are limited, and operators rarely attempt sophisticated operations such as retrieving and utilizing tools, deploying sensors, or building up world models. This project has focused on methods to utilize this video information to enable monitored autonomous behaviors for the mobile manipulator system, with the goal of improving the overall effectiveness of the human/robot system. Work includes visual servoing, visual targeting, utilization of embedded video in 3-D models, and improved methods of camera utilization and calibration.

Acknowledgements

The work presented in this report represents a summary of work conducted in the last three years, with an emphasis on the most recent year's work. Many Sandians and Sandia contractors contributed to the work presented within this report and should be acknowledged. Hanspeter Shaub and Chris Smith developed much of the visual snake algorithms used. Eric Gottlieb, Fred Oppel and others invented the Umbra software environment that was heavily utilized throughout this project. Dan Small developed the live video input capability within the Umbra framework. Mike McDonald generated much of the original Active Sketch code, based off of targeting code developed by Chris Wilson. Michael Saavedra developed the zoom camera housing and helped to keep much of the hardware running. Scott Gladwell developed planners that converted active sketch commands into robot actions. Phil Bennett provided the problems, the motivation and the customers to keep us focused on real tasks.

CONTENTS

1.0 Introduction

Mobile manipulator systems, such as the Remotec Advanced Manipulator (Figure 1) are used by emergency response personnel to respond to threats posed by improvised explosive devices (IEDs). Existing commercial systems are limited in their autonomous capability because they rely solely on operator interpretation of the live video images and direct operator control of the manipulator, i.e., teleoperation. With this LDRD we have focused on developing techniques to reduce the burden on the operator by digitally sampling the live video image and developing algorithms that use this information to semi-autonomously control the remote manipulator system.

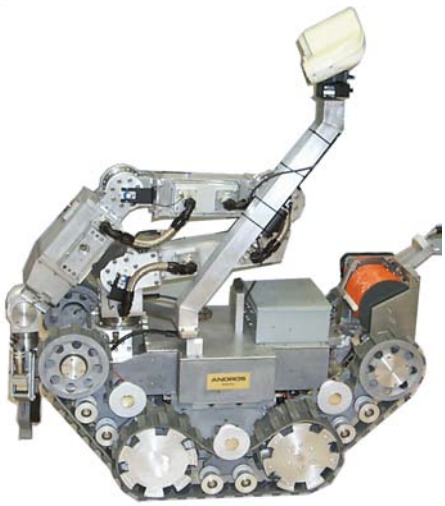


Figure 1: Remotec Advanced manipulator with target head

The mobile response systems that have been considered, are all similar in setup. A remote vehicle is operated out of direct line of sight of the operator. A live video feed is brought back to an operator console similar to one shown in Figure 2 and captured on a frame grabber attached to the operator control unit. Snapshots of this video can be taken, and/or the live video stream used directly.

A large number of tasks related to the utilization of video in the remote control of emergency response vehicles have been conducted during the course of this LDRD. Automatic tool pickups have been demonstrated using visual servoing algorithms based on statistical pressure snakes. The algorithms have proven to be robust despite the unstructured lighting conditions. The remote manipulator is able to visually servo based off of the snake inputs to automatically grasp tools. This work is described in section 2.0.

Visual servoing uses the continuous live video feed from the wrist camera. Visual targeting, on the other hand, utilizes a pair of snapshots from a pair of calibrated cameras. Visual targeting and active sketch technology have also been further developed within this project. With the active sketch framework simple primitives are used in a pair of stereo images. Selecting these primitives enables a set of possible actions for the operator to choose between. The latest active sketch implementations are described in section 3.0.

Figure 2. Remote Manipulator Operator Console

Embedded video provides context for live video information and allows a correlation between live video data and virtual representations of the world. Work in embedded video for mobile response system is described in section 4.0.

Camera calibration is crucial for converting image data to robot action. Section 5.0 documents developments in camera calibrations, including the latest developments utilizing calibrated zoom camera systems.

2.0 Automatic Tool Pick-Up With Visual Servoing

An emergency response robotic operator can deploy a wide variety of tools, depending on the task at hand. Grappling hooks, explosive tools, X-ray equipment, chemical sniffers, small drills etc., might all be utilized. In many cases, these may be picked up from a trailer, or from a secondary pack-mule robot, and the location of these tools with respect to the robot cannot be precisely known. Figure 3 illustrates the retrieval of tools from a pack-mule robot.

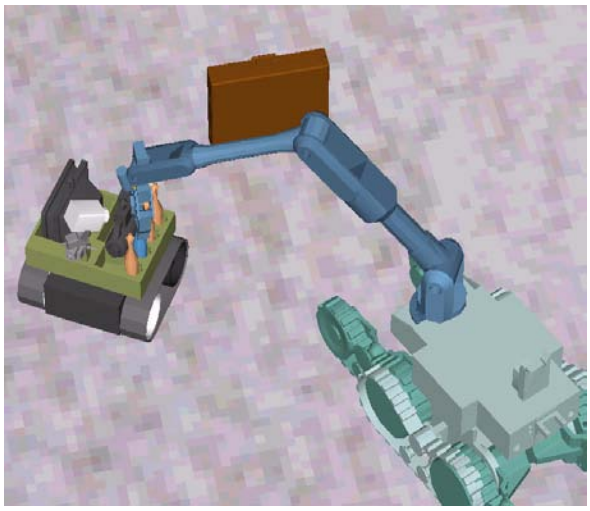


Figure 3: Pack-Mule Retrieval of Tools

Teleoperating a tool grasp can be a slow procedure, since the position and orientation of the tool must be controlled precisely to prevent damage to the tool or the tool holder. The tools themselves, however, can be engineered for easy recognition and retrieval. Tool grasps, are therefore, an excellent candidate for automation. In this section we describe how we have utilized statistical pressure snakes and visual servoing to demonstrate automatic visual grasping for telerobotic systems.

2.1 Statistical Pressure Snakes

Our approach to visual servoing utilizes statistical pressure snakes, based on the Perrin-Smith model [1]. A pressure snake is a mathematical construct representing a closed loop segmented contour on an image plane. The snake's length and position is constantly being pushed by image pressures, as it seeks to find a minimum energy solution. Regions of the image with a "good" feature, such as its measured closeness to a desired color generate pressures that push out the snake from its interior, while contour tension tries to collapse the contour. Snakes can be initiated by either starting at an initial selection point and growing the snake outward, or by using the entire region and collapsing the snake around a region. In either case when the snakes are well tuned, they should expand or contract to follow the contours of a shape of interest after a few iterations.

Snakes provide many advantages over other image processing techniques. First, the algorithm is fast. The computations are typically of order n, where n is the number of segments in the snake. This number can shrink or grow depending on the overall length of the snake contour, but is still small compared to the image size. Only a small window of pixels around each vertex is all that is used for image analysis in a given iteration. Second, the snake allows for easy interaction with the user. A simple button click can define a seed-region for growing a new snake. The evolution of the snake algorithm is easily displayed by overlaying a snake drawing on the live video, so the operator can monitor operations in real-time. The snake data structure can be queried for essential object features such as center, size and orientation, which can be used to drive the visual servoing operations.

The development of color based statistical pressure snakes under this LDRD has been well documented in number of SAND reports and conference proceedings. [2-7]

2.2 Using Snakes For Tool-Pickups

Snakes help to define a region of interest for the robot system, but do not directly result in any useful operations. To become truly useful for the mobile manipulator system, information derived from the snakes must be utilized as one part of a motion planning algorithm. In a completely unstructured environment this is difficult, since the robot system has no method to transform general object contour information into a plan of action. By constraining the problem and controlling a part of the environment, however, a automated activity for the mobile manipulator can be developed. This has been done for tool pickups.

Industrial robots commonly pick up tools without visual servoing simply by moving to a series of pre-taught points. Mobile manipulator systems, on the other hand, typically lack the structure and the precision required to pick up tools open loop. In many cases tools pickup locations are improvised, tools are placed on trays or on tool racks without precise alignment locations. Tools may be delivered off of a mobile pack-mule, or retrieved from the back of a tool trailer, without any precision a priori location information available to the robot. Mobile manipulators tend to be less rigid, with more mechanical play and less accurate joint calibration, so even if the tool position is precisely known, an open loop attempt to grab a tool may fail even if the tool location with respect to the robot is precisely known. Industrial robots also work in a workcell where humans, and their corrupting influences are avoided. Tools are always located in the same locations, because no-one was allowed to move them away. Emergency response robots, however, work closely with human operators, and are thus subject to the whims of operators to move and deploy different tools.

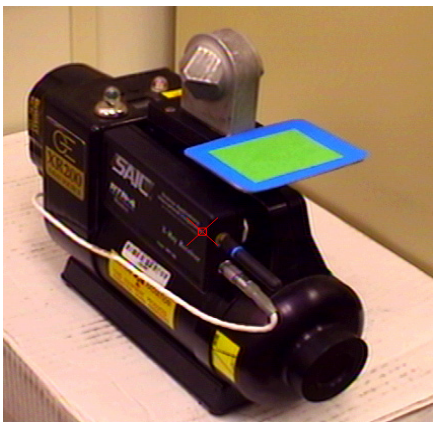


Figure 4: Standardized Tool Pickup on an X-Ray source tool

To enable automated tool pick-ups, we add a level of structure to the tools themselves. In particular we place a colored rectangular patch at a known position and location offset from the tool pickup point. This patch is designed so it will be completely viewable from the gripper camera during the entire approach sequence. Figure 4 shows a colored patch attached to a telemanipulator tool, in this case an X-ray source. By standardizing the distance and size of the rectangular patch, we can directly convert the information gleaned from the active snake, to a motion plan to pick up a tool. Since the size of the rectangular patch is known a priori, the measured area in pixels of the enclosed snake contour maps directly to an absolute distance to the target. The principal axes of the rectangular block can be used for orientation alignment, and the center of the target is used to determine the amount of translation needed. Thus a single rectangular block can be used to determine four degrees-of-freedom of motion: *x, y, z* translation, and roll orientation in the image frame. In addition, by controlling the color of both the rectangular patch and its background, an extremely robust snake can be obtained.

Pitch and yaw errors in the image plane for the rectangular block also result in skewing of the sides of the rectangle, but the measurement sensitivity to pitch and yaw is far less than for the other parameters and was not used.[1] In practice, tools are oriented so that a standard vertical approach

is needed to make the grab, and no additional orientation alignment other than roll is needed. Only if the robot is operating in very hilly terrain and neither the robot nor the tool caddy can be placed on the same slope is this seen as a problem.

2.3 Improving Tracking Reliability

Emergency response operators have been reluctant to utilize autonomous manipulator behaviors, because it is easy for something to go wrong in an unstructured environment. It is difficult to replicate the operator's ability to map camera imagery into a course of action for the manipulator. Even for the snake based algorithms, changes in lighting conditions, occlusions, motion blur, auto-iris adjustments, etc., can cause the snakes to perform unpredictably. For operator acceptance, however, the algorithms implementing autonomous behaviors need to be fool-proof. Luckily for telerobotic systems, there is always a fallback course of action. If the autonomous system cannot perform the operation and can recognize it cannot perform the operation, the operator can always be asked to intervene. Thus the key is to recognize when the snake has failed in some way to track the desired shape, and then halt the autonomous behavior and request assistance from the operator.

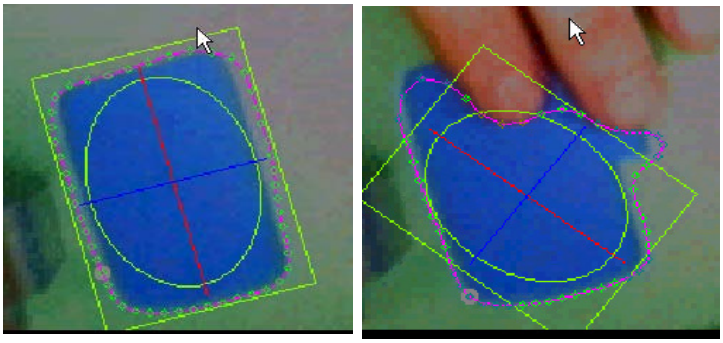


Figure 5: Rectangular Snake with Boundary Checks

To help insure the quality of automatic tool pickup we have implemented a simple sanity check. A snake tracking a rectangular block will appear as a rounded rectangle. The finite number of points and the line pressure will result in the corners being rounded off. The amount of rounding depends on the number of points in the snake, the image pressure, and the size of the contour. In any case a properly performing snake should always be bounded by a slightly larger rectangle on the outside, and by a slightly smaller ellipse on the inside. If any points in the snake are driven outside of these bounding areas, then the visual servoing is stopped, and the operator's assistance is requested. In addition to this sanity check, a number of additional checks regarding acceptable size, aspect ratio, and rate of change are also performed on the active contour.

Figure 5 shows the snake and the boundary regions during normal operation and when corrupted by occlusion. The points of the snake outside the bounding regions are flagged accordingly.

2.4 Implementation Details

The visual servoing application was implemented using two Sandia developed technologies, Umbra and SMART [8-11]. Both are connector based technologies that allow the developer to

1.To improve sensitivity to pitch and yaw a dual snake system could be used, whereby a 3-dimensional feature such as a peg or elevated block is placed in the center of the rectangular block. A snake is used to capture the outside contour, and the contour of the elevated peg surface, and the centers of the two snakes are computed. Any offsets in the centers is then directly converted into pitch and yaw errors.

combine system modules to implement behaviors. Umbra is focused on graphics, visualization, and operator interaction. SMART is used for real-time servo control and teleoperation. Together, they provide a rich environment for advanced robotic systems.

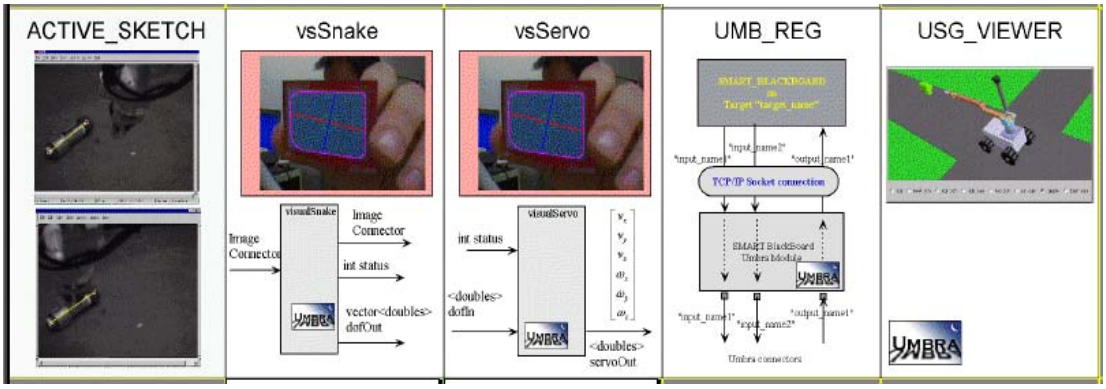


Figure 6 Umbra Modules Used for Implementation

Figure 6 shows the icon representation of the relevant Umbra components in the system. An ACTIVE_SKETCH module provides a connection to the live camera frame grabber (via a DigiCam Umbra module). The *vsSnake* module uses the image connector as an input and implements the snake algorithm. The *vsServo* module takes the snake inputs and generates a velocity error command to drive the robot's tool. The *UMB_REG* module provides a direction socket connection between Umbra's connectors and SMART's data registers, allowing the servo error command to directly drive the embedded robot controller. Finally, the *USG_VIEWER* is used to embed an Umbra scene graph visualizer within the application.

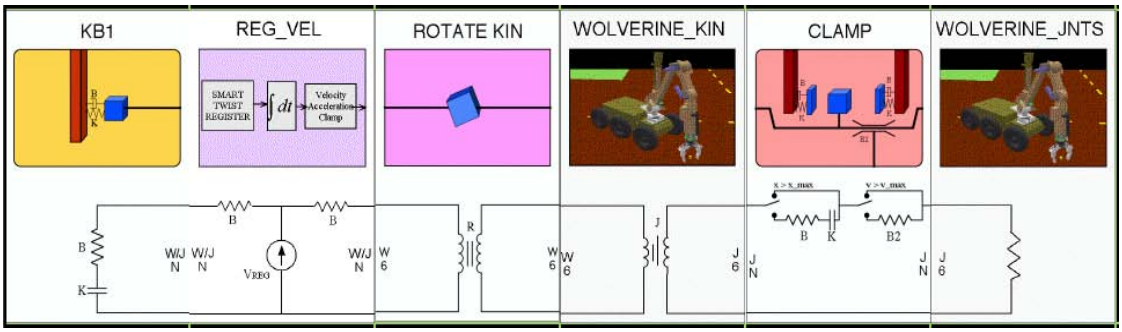


Figure 7 SMART Modules Used for Visual Servoing Implementation

Figure 7 shows the icon representation of the SMART modules used in the system. The *REG_VEL* receives the velocity error commands from the *UMB_REG* module. The *ROTATE_KIN* module rotates the velocities into the tool reference frame, the *WOLVERINE_KIN* converts tool speeds into joint speeds, and the *WOLVERINE_JNTS* module drives the robot.

2.5 Stages of Automated Visual Servoing

The visual servoing process for an automated tool grab is illustrated in Figure 8. First the robot is moved to a point above the expected location of the tool. Initial snake acquisition is obtained by either the operator clicking on the colored patch on the tool and growing the snake outward, or by shrinking a snake around the entire region until it collapses on the tool region of interest. Once a valid tool patch has been acquired, the visual servoing stage begins. During visual servoing the 4-

axis correction values are fed from the *vsServo* umbra module to the *REG_VEL* module on the SMART controller. Motion proceeds until either the goal pose has been reached, or the snake has failed to maintain its shape for some reason.

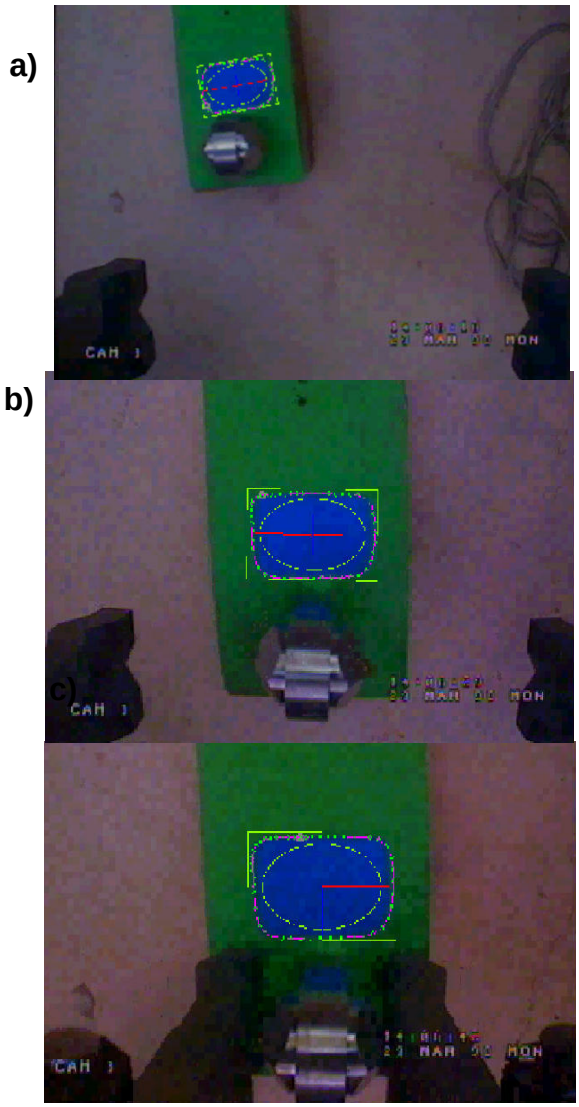


Figure 8: Visual Servo Grasp: a) Initial Acquisition; b) Approach; c) Final Grab.

3.0 Visual Targeting and Active Sketch

Active Sketch and Visual Targeting have been the mainstay technologies for developing autonomous behaviors for mobile manipulators. Although they were invented within prior research projects, they have been heavily utilized, developed and refined during the course of this project. In this section we define these approaches and document their current implementation as applied to mobile robot control.

3.1 Visual targeting

Visual Targeting provides the basis for the more operation focused Active Sketch techniques, and has been utilized by robotics researchers for years [13] Targeting requires a pair of calibrated camera snapshots to be taken of a common object viewed from different angles. Typically this is done using a calibrated stereo camera head, with snapshots being taken simultaneously, but this isn't always the case.

As an example, Figure 9 shows a Sandia developed stereo targeting head containing two fixed focal length cameras mounted on a pan-tilt-unit, which is attached to a mast that is in turn attached to the robot torso. The operator moves the PTU with a joystick to point both cameras at a target of interest, and then clicks a button to take a pair of snapshots.

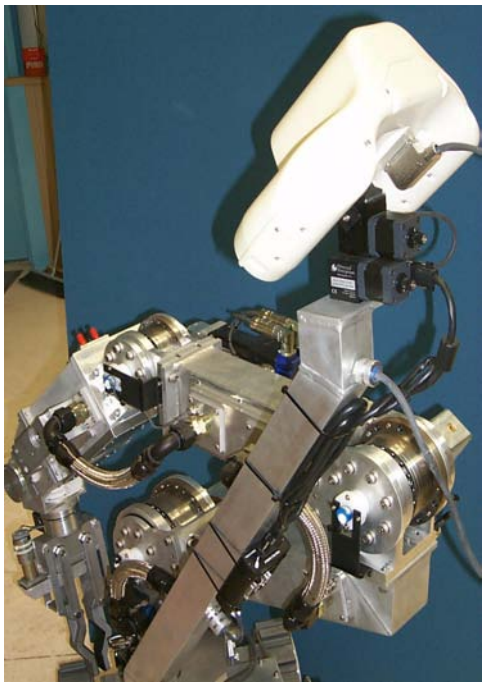


Figure 9: Visual Targeting Head on a Remotec Advanced Mobile Manipulator

The user then selects a common feature in both images. Using the mathematics of the calibrated camera model, each pixel selected in space corresponds to a vector cast in 3-dimensional space. The pair of vectors from the two images correspond to a unique point in 3-D space, ideally the point of intersection of the two vectors, but more typically due to image and calibration errors the point of nearest intersection of the two skew vectors. In summary, visual targeting simply uses triangulation of two calibrated camera images too determine the 3-D position of a point in space with respect to the robot. And a feature so selected is considered "targeted".

3.2 Active Sketch

Visual targeting in itself does not specify robot action. All it specifies is the 3-D point of a feature in the robot's workspace. For a robot to perform an automatic action, however, much more information must be available. A robot task requires much more than the 3-D position of a feature in space. It requires a complete goal pose, consisting of a desired 6-DOF position and orientation of the robot's gripper. It also requires an approach path, the series of joint angles that will safely take the robot from its current location to the goal pose.

The Active Sketch technology was developed at Sandia, [14,15] to bridge the gap between 3-D position targeting and robot action. With Active Sketch, the task is divided into two steps, the object specification and the task specification. In our implementation, the available object specification tools are simple geometric primitives: a point, a line, two connected lines representing a plane, and three lines with a common vertex used to define a volume called a triad. Selecting the tool places a copy of the geometric primitive into both images, where they can be dragged to align with common features in the image plane.

Rather then forcing the operator to completely match image features at each vertex, however, Active Sketch, utilizes the epi-polar constraint to substantially simplify the stereo matching problem. The epi-polar constraint is a realization that the ray cast by selecting a pixel in a target image, corresponds to a line when viewed from a second range image. By limiting mouse selections in the range image to points that lie along the constraint line, the search process becomes a one dimensional selection rather than a two dimensional selection. Not only is the likelihood of the operator selecting the wrong feature substantially diminished, it is also possible to find alignments even when targets are completely occluded in the second image.

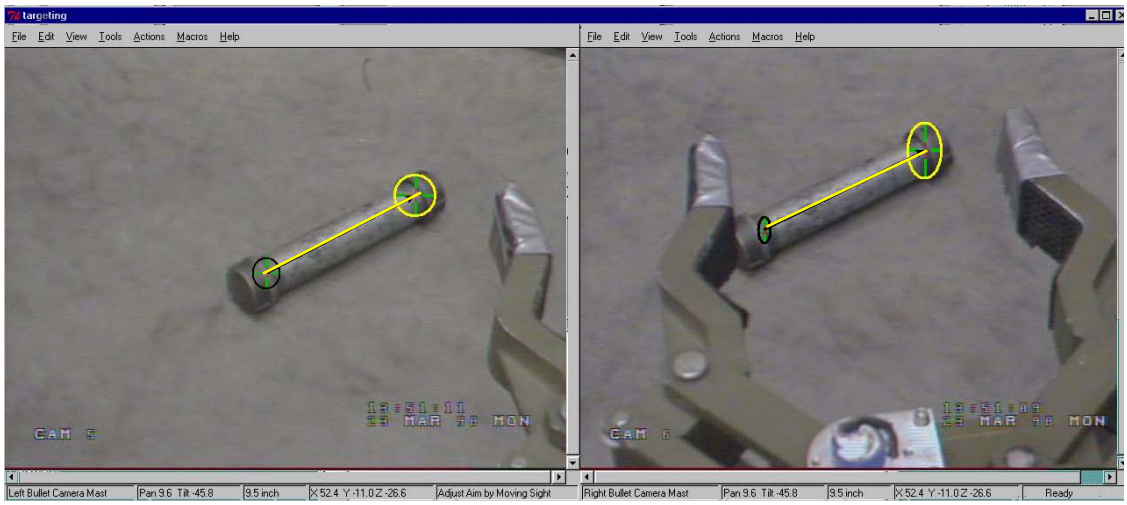


Figure 10 Stereo Snapshots of a pipe.

One of the object specification tools are placed in the image plane, e.g., point tool, line tool, plane tool, and triad tools, and then moved in alignment with common features in both images. Selecting and highlighting the object makes available a set of actions. An intelligent planner then uses the 3-D location data as a starting point for planning a path to execute the desired action. Figure 10 shows a pair of stereo images taken within Active Sketch. A line tool if being used by the operator, and has been dragged over the features of interest, namely the axis of the pipe bomb.

3.3 Implementing Automatic Motion Behaviors

Figure 11 below shows a typical trajectory task control panel used for mobile manipulation. It works intimately with the Active Sketch environment. As different active sketch primitives are chosen, i.e., points, lines or surfaces, different motion macros become available. When no features have been selected only the fixed, “Canned Motion” routines are allowed. If a line tool is chosen, macros such as “Grab Pipe” or “Move Around Axis” become selectable.

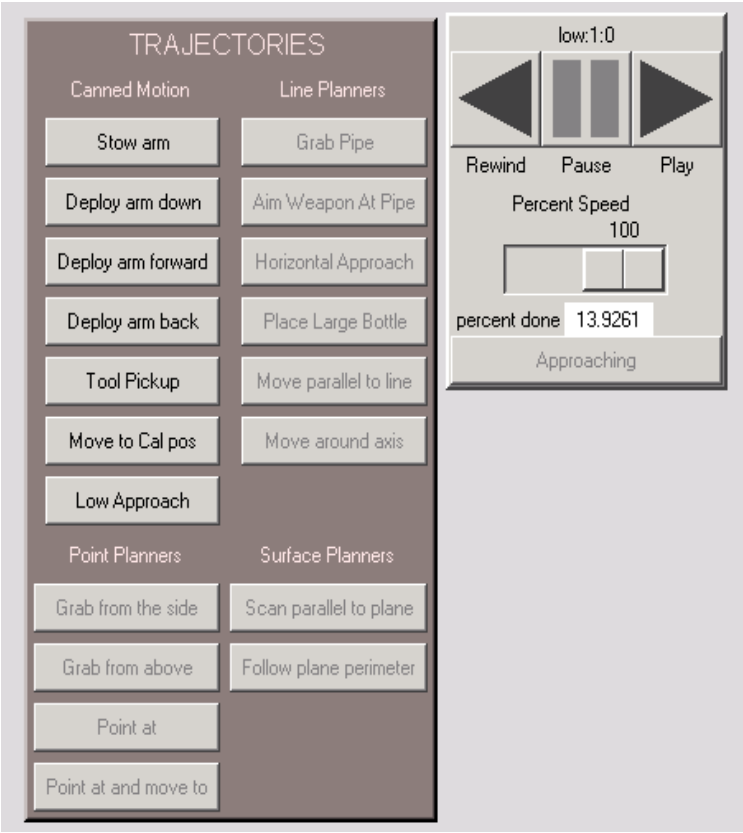


Figure 11 Automated Control Panel

The automatic control panel also includes a large "VCR" controller. Play begins automatically when a trajectory macro has been selected. The operator is free to pause, backup, change speeds, and resume motion as needed, simply by clicking on the interface buttons.

For example, to grab a pipe, the operator first aims the stereo camera system at the pipe of interest, and takes the pair of snapshots as in Figure 10. A Line tool is selected and dragged across the image. Once the line is selected a series of line-based trajectory macros becomes available to the operator, as in Figure 12. By clicking on the appropriate button, the "Grab Pipe" motion planner is called, and a path to grab the center of pipe from above is planned and initiated. The cameras are automatically slewed over to directly monitor the operation and the robot starts to execute the planned motion. The operator monitors the motion, and if needed, can pause or add position corrections.

A door opening task can be conducted in much the same fashion. In this case the line of interest is the hinge line of the door and the "Move Around Axis" button should be selected. This planner will apply the right hand rule to the axis and sweep out an arc around the axis, starting with the robots current grip point.

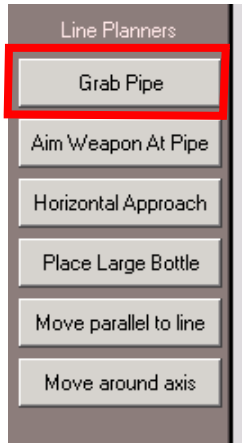
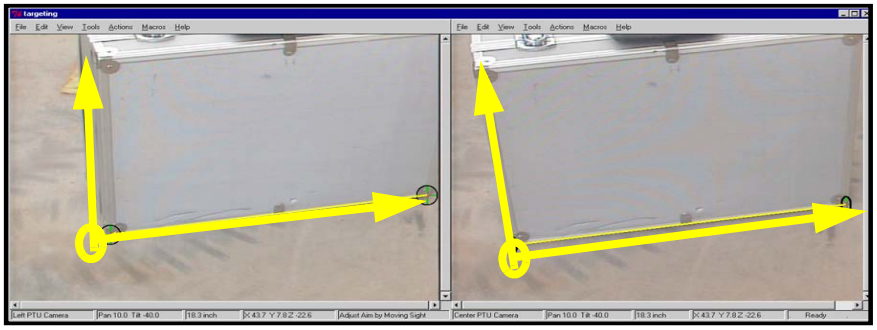


Figure 12: Operator Line Tool Macros

Figure 13 Automated Door Opening

Other line planners will approach a designated line from the side, move parallel to the line, aim a weapon along the axis of the line, or assuming the line represents the bottom of a package, will place a water bottle disruption device directly in front of the package.

A number of operations can also be performed on the surface of an object. Chemical sniffers need to be swept across the surface of an object in order to pick up trace compounds. By using the Active Sketch Plane tool a rectangular surface area can be designated in the model. Selecting the plane tool enables access to both the surface scan and follow perimeter planners. (Figure 14).



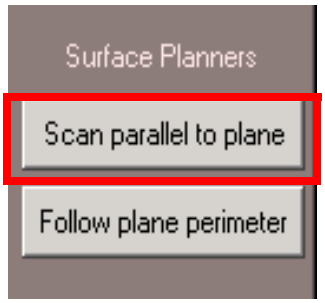


Figure 14 Automatic Surface Following

4.0 Embedding Visual Information

Another problem inherent in the utilization of remote cameras is the tunnel vision phenomena. Operators lose spatial awareness of the live video feed and are often unable to correctly interpret imagery. By embedding visual information within a 3-D modelling environment this problem can be largely averted.

Figure 15: Embedded Visual Model

During the course of this LDRD a number of visualization tools have been added to the system. Live camera imagery has been embedded in correspondence with the graphical model, and video imagery is cast onto an embedded video screen. The operator is able to change the level of transparency of the live video feed to move between live views and virtual views as needed.

Image snapshots can also be taken and embedded within the working model. Each snapshot has a working image frustum associated with it which can be visually displayed. The images can be viewed from the original camera position and orientation, or from any other location. Figure 15 shows a pair of snapshots taken of a box from a different points of view.

The active sketch tool can be used to add geometry to the scene graph. By using the triad tool, box shaped objects can be readily placed and visualized within the virtual world. This not only helps keep the operator informed of objects in the robot's workcell, but it also serves as a forensics tool, to help build up a working model of the robot's environment. Figure 16 shows a box that has been added to the scene graph via the active sketch interface. The semi-transparent image lines up with the 3-D geometry when viewed from the camera's point of view.

Figure 16 Semi-Transparent Video overlay over a modeled box

5.0 Improving Camera Calibration

For the robot system to be able to perform operations semi-autonomously, precise 3-D measurements must be directly obtainable from the video feedback, and correlated from multiple camera views. This task is called camera calibration, and involves determining the mapping between image plane coordinates and 3-D space. Good camera calibration is crucial for visually directed mobile manipulation systems. Appendix A summarizes the terminology and algorithms used for computing the calibration of a fixed focal length camera. Appendix B describes the decomposition of the camera calibration matrix into intrinsic and extrinsic representations.

Mobile manipulator systems typically have a suite of cameras: a drive camera, a reverse camera, one or two wrist cameras, and a torso mounted pan and tilt targeting head. For visual targeting systems, camera calibration has typically involved accurately calibrating a stereo pair of fixed focal length targeting cameras. These cameras are taken to a precision optical bench, where both intrinsic and extrinsic camera parameters are measured and recorded. The calibrated stereo camera system is then attached to the robot, and the position offset is measured.

Unfortunately, our experience with applying calibrated camera heads to mobile manipulation has uncovered a number of problems with this approach. Off-line camera calibration works well in a lab environment, but in the field, cameras are too easily jostled and moved from initial setup positions. Small orientation alignment errors can lead to relatively large targeting errors. Camera mount points may change or degrade over time. All of this leads to error buildup for targeting operations.

The single stereo head is also problematic. To fit within the footprint of the robot, the spacing of stereo cameras is typically ten inches or less. This small spacing reduces the accuracy of distance measurements. A torso mounted camera may also have issues with arm occlusion. The arm often needs to be rotated out of the way for the targeting operations to succeed, reducing the both the operational speed and advantages of autonomous operations. In addition, remaining cameras, such as drive, reverse and wrist cameras, are under-utilized. These cameras could provide valuable information from a different perspective, if calibration for randomly mounted cameras were straight-forward.

Camera calibration has, to date, relied on fixed focal length cameras. Unfortunately without zoom capabilities, these cameras have limited utility. They work well for targeting operations on objects with measurable features between 4” and 20” in length, but are unable to adequately target small features such as key-holes, or large features such as shipping crates, packages or buildings.

This research project has begun to address these issues, but the work has not been completed. We have developed an interactive tool for calibrating cameras using a benchtop fixture. We have developed a new algorithm for pose estimation which should ultimately allow improved in situ camera calibration. We have partially developed an interactive graphics tool for pose estimation which makes it possible to use any known geometric object as a possible calibration tool. We’ve begun to integrate calibrated zoom cameras into our environments and have developed a calibration approach for these cameras. This section describes these most recent developments.

5.1 Intrinsic Camera Calibration and the Distortion Model

There is a significant body of work in camera calibration [16-18]. Typically camera calibration can be divided into three stages, intrinsic and extrinsic calibration, and pose estimation. Intrinsic calibration involves the measurement of camera focal lengths, camera distortion, and pixel offsets.

These characteristics are fundamental to a particular camera/lens combination and can be done off-line on a precision bench top without a problem. For our intrinsic model we are using the model used within the Intel® Open Source Computer Vision Library camera code [19]. It is a collection of C functions and few C++ classes that implement some popular algorithms of image processing and computer vision.

The lens distortion model is parameterized as follows. Let (*u,v*) represent the ideal pin-hole projection model parameters and let $(\tilde{u}, \tilde{v})$ represent the measured distorted values. Similarly let (*x, y*) represent the ideal distortion free parameters and $(\tilde{x}, \tilde{y})$ represent the distorted physical model.

Taking into account the first two expansion terms in a radial distortion model gives,

$$\begin{aligned} \tilde{x} &= x + x(k_1 r^2 + k_2 r^4) + 2p_1 xy + p_2(r^2 + 2x^2) \\ \tilde{y} &= y + y(k_1 r^2 + k_2 r^4) + 2p_2 xy + p_2(r^2 + 2x^2) \end{aligned} \qquad (1)$$

where, $r = x^2 + y^2$. Since $\tilde{u} = f_x\tilde{x} + u_0$, $\tilde{v} = f_y\tilde{y} + v_0$, $u = f_x x + u_0$, $v = f_y y + v_0$ this results in

$$\begin{aligned} \tilde{u} &= u + (u - u_0)\left(k_1 r^2 + k_2 r^4 + 2p_1 y + p_2\left(\frac{r^2}{x} + 2x\right)\right) \\ \tilde{v} &= u + (u - v_0)\left(k_1 r^2 + k_2 r^4 + 2p_1 x + p_2\left(\frac{r^2}{y} + 2y\right)\right) \end{aligned} \qquad (2)$$

These latter two relations are used to undistort the images from the camera using the OpenCV routine UnDistort. The camera's distortion is modeled by four coefficients: two radial distortion terms, k_1 and k_2, and two tangential distortion terms, p_1, p_2. In practice, these parameters are obtained either by taking multiple snapshots of a checkerboard and letting an OpenCV routines compute the coefficients, or by running the UnDistort OpenCV function on the image and manually adjusting the parameters until straight lines on a fixture map to straight lines in the image. Unfortunately, for cameras with relatively small distortion, the native OpenCV often fail to converge, and only the manual adjustment method is available.

The distortion model uses second and fourth order quadratic terms to represent distortion in the system. Once distortion parameters are estimated, image distortion can be reduced by running the image through a image distortion correction tool. The distortion filter allows us to decouple image distortion from camera calibration. By first running unDistort on any captured image, all subsequent calibration measurements can be assumed to operate on an ideal pin-point camera image. The subsequent discussion assumes that this distortion correction has been applied.

5.2 Extrinsic Calibration and the Camera Calibration Tool

Extrinsic calibration measures the position and orientation of the camera with respect to a fixed reference frame. It is essentially the same as pose estimation, except for two important differences. It is the transform that is closest to the camera mapping, and is typically computable using a benchtop setup. Pose estimation, on the other hand, is done on site. In addition, for a zoom camera the extrinsic calibration transform will also change as a function of zoom settings, since the

effective center of the camera model changes as a function of zoom. The pose estimation, should be constant as a function of zoom setting.

Figure 17 shows a interface tool that has been developed to calibrate cameras using a bench top fixture. Both the intrinsic and extrinsic parameters of the cameras can be calibrated with this tool, but not the final camera pose estimation. The intrinsic parameters can be computed in one of two methods: the native method within OpenCV which takes a series of pictures of checkerboards and computes distortion parameters, and the fixture based method which computes the pin-hole camera model intrinsic parameters, and allows the distortion to be determined interactively.

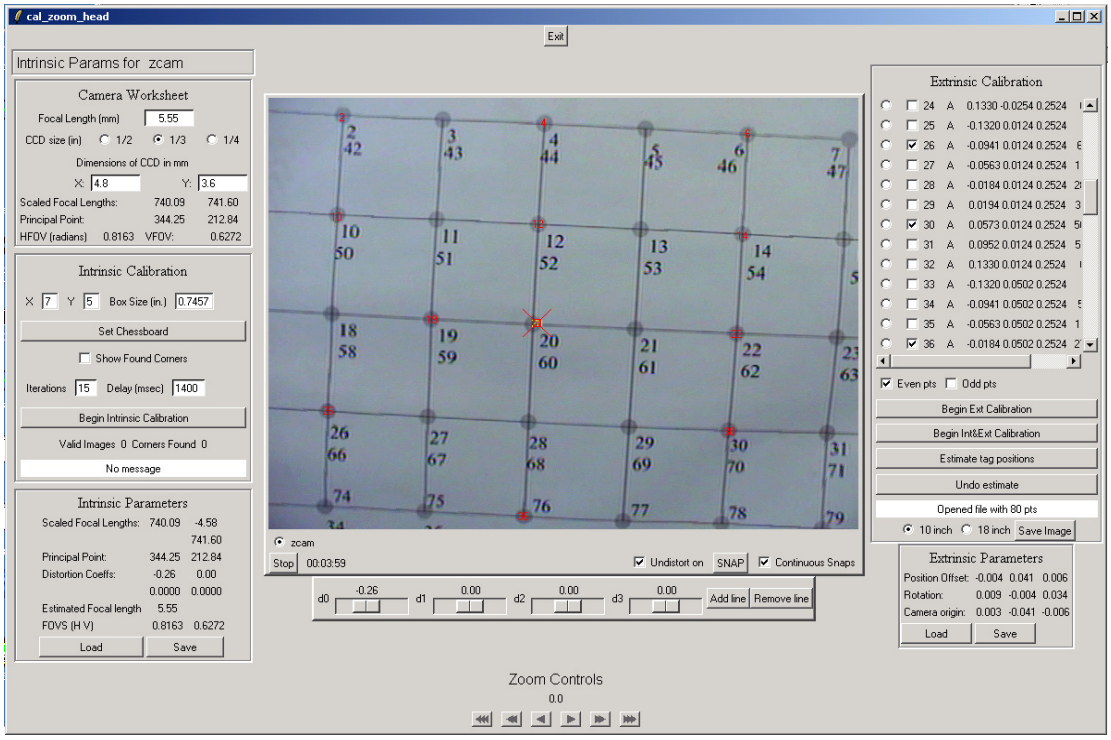


Figure 17 Fixture based Camera Calibration Tool

The method requires a camera fixture containing a series of measured points. The points must not all be co-planar. In practice we have used either a series of posts on a precision optical bench, or a calibration fixture with a sliding panel containing a grid of known points. (see Figure 18)

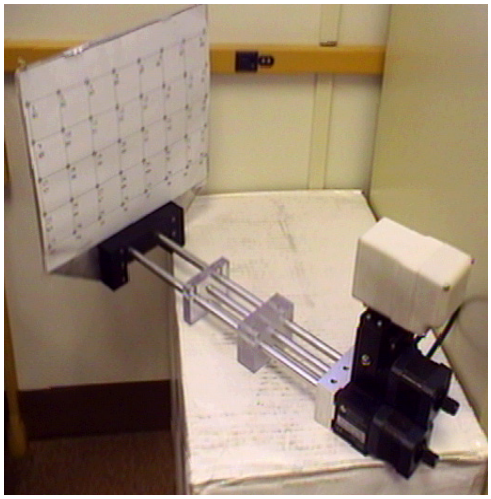


Figure 18 Calibration Fixture with Adjustable depth feature.

5.3 The Non-Perspective Pose Estimation Problem

A camera can be carefully calibrated on a bench top, and then fail when placed on the robot due to inaccuracies in mounting the camera system. For visual targeting to work well, this last transformation needs to be computed accurately. This transform is called the camera pose, and consists of the position and orientation of the image device with respect to the robot. The same problem arises when the position and orientation of known objects need to be known with respect to the robot's position.

Pose estimation focuses on just the orientation and position of a device, either a camera or targeted object. In many cases, the imaging device is a single perspective camera with a single snapshot, with multiple features selected. In this case (assuming the pin-point model of the camera) all of the UV selections correspond to rays which all intersect at a common point. If three such well-determined points are selected, then a correct solution can be found algrebraically by solving a polynomial. In general, however, four possible solutions exist, and additional information must be used to determine the correct solution. This problem is called the P3P problem.

To improve the accuracy of the P3P solution a number of researchers have extended the results to N points ($N>3$). By using the 3 point solution as an initial seed, further refinement and accuracy is obtained by adding more points to the system. The fourth point is often used to resolve the multiple solutions found from solving the algebraic equation. All points must still intersect at a common point, however, for these approaches to work.

In tracking operations a target object may have essentially the same perspective distortion over all feature points. In this case, a weak perspective assumption can be made, and the equations can be simplified substantially. The POSIT algorithm is a highly efficient approach for computing camera frames for perspective systems using a weak perspective assumption.

A number of general imaging devices (GIDs) don't fit the perspective camera model, i.e., when the ray segments don't all intersect at a common point. This may occur with high distortion lenses, line cameras, or in our case, due to obtaining pose estimation from multiple camera views, from a

camera mounted on a PTU system. The non-perspective n point problem of finding a camera object pose is a variation of the PnP (Perspective N Point problem), without a common intersection point [20]. Finding an effective means of solving this problem is critical for robot pose estimation for our mobile manipulators.

In general, the problem amounts to finding the rotation matrix and translation offset between the camera system and the object, by establishing a correspondence between objects in the image frame (i.e., *u, v* coordinates) and objects in a world frame (*x, y, z*) coordinates. The rotation/translation can be represented by 6 independent parameters, and thus by defining 3 UV-XYZ correspondences a solution for the rotation frame should be attainable.

In Appendix D we develop an algorithm for iteratively solving the *nPnP* problem assuming a good approximate value for the rotation and offset has already been obtained. The algorithm is generic and can be used to find any unknown transformation given a sufficient number of image correspondences.

5.4 The Calibration Transform Loop

A generic solution to camera and object calibration uses more then just a intrinsic and extrinsic model for the camera. Cameras may be attached to moving devices with their own kinematics. Objects are typically defined with respect to a world coordinate frame that is independent of the camera frame.

To accommodate all of these possible transforms, we will augment the conventional camera transformations with some additional transforms. Let C_{int} and T_{ext} represent the conventional camera intrinsic and extrinsic transformations. These can be computed using the benchtop calibration approach of section 5.2. T_{kin} represents the kinematics of an aiming device, typically a PTU or the kinematic chain of a robot, and is assumed known, but may vary as a function of joint angles. T_{dev} represents the transform of the image device with respect to a world coordinate system. T_{obj} represents the transform of a object target with respect to a world coordinate system.

The equation relating all of these transformations is given in Equation 3, and is called the calibration transform loop. It relates image plane coordinates identifying features in a snapshot of a imaging device to the position of points on the target object.

$$\begin{bmatrix} U_i \\ V_i \\ t_i \end{bmatrix} = C_{int} \begin{bmatrix} & & 0 \\ I_3 & & 0 \\ & & 0 \end{bmatrix} T_{ext} T_{kin} T_{dev} T_{obj}^{-1} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \qquad (3)$$

This transformation loop is shown graphically in Figure 18. The calibration objective is to find all of the transformation matrices in the system by a combination of benchtop component calibration and in-situ pose estimation.

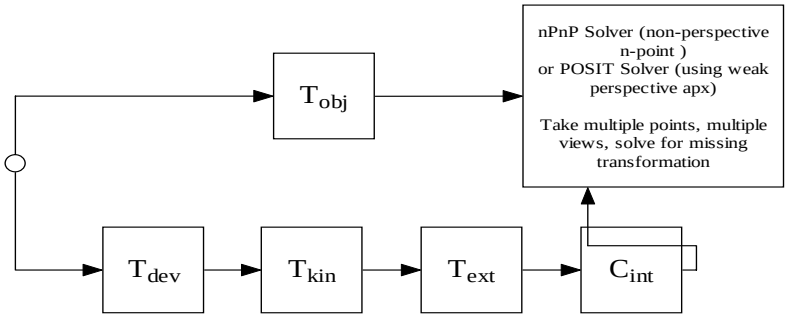


Figure 19 Camera Transforms for Final Pose Estimation

5.5 Developing an Interactive Pose Estimation Tool

An interactive camera pose estimation tool is being developed within an Umbra based graphical environment that utilizes the *nPnP* algorithm. The objective is a tool which can both accurately place an imaging device in the environment given known 3-D geometry (T_{dev}), and place 3-D geometry given known camera locations (T_{obj}). Rather then relying on a calibration fixture, the pose estimation tool is being designed to use any known CAD geometry, and to allow the user to pick the feature points interactively. These features are called calibration points, and represent any feature that can be easily recognized from visual inspection of an image. To fix the relative fixed-body pose between the camera frame and the world frame, a minimum, of three calibration points must be found. The user can define calibration points within the Umbra environment simply by clicking on features. The system will automatically attach a point to the surface of the object under the mouse pointer at the point of intersection of the ray cast, and compute its 3-D position in the model. Figure 20 below shows a camera image embedded within a 3-D reconstruction and four calibration points determined interactively by the user.

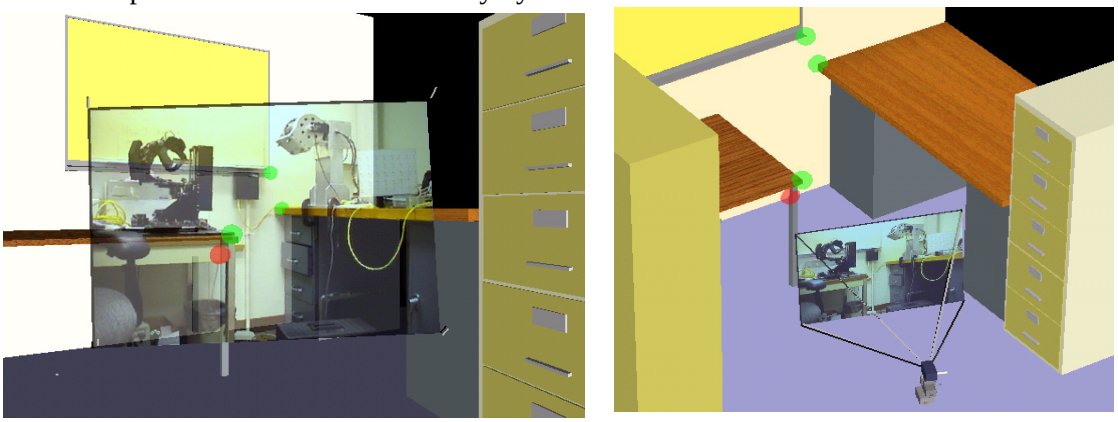


Figure 20 Calibration Points within a 3-D Environment

Once a series of calibration point correspondences have been made, the user of the calibration tool will be able to calibrate any one of the following: the transform for the calibration object with respect to the camera (C_{obj}), the transform of the camera with respect to the device (*Cext*), the cal-

ibration of the camera device, (T_{dev}), or the intrinsic parameters for the camera. The final transformation, the internal kinematics of the camera device, (T_{kin}), is assumed known.

A minimum of three correspondences should be taken, but more is desirable to improve accuracy. There may only be a single image with lots of UV correspondence points, or there may be lots of images, with only a single correspondence point on each image, or a mixture of both. For instance, to calibrate a drive camera with respect to a robot, a series of individual snapshots must be taken with the robot holding a robot tool. To best determine the T_{obj} transform tor a PTU system, on the other hand, a series of images from widely varying pan and tilt angles should be taken to optimize orientation accuracy.

A preliminary version of the Camera Pose Tool is shown in Figure 21 below. The tool allows multiple snapshots to be taken of an environment. For each picture correspondence between the 3-D calibration balls and image features are determined interactively by the operator. A desired object is selected which should have a consistent pose in all images. The graphical representation gives the starting position and orientation for the iterative pose estimation algorithm. The pose is then accurately determined by executing the algorithm described in Appendix D.

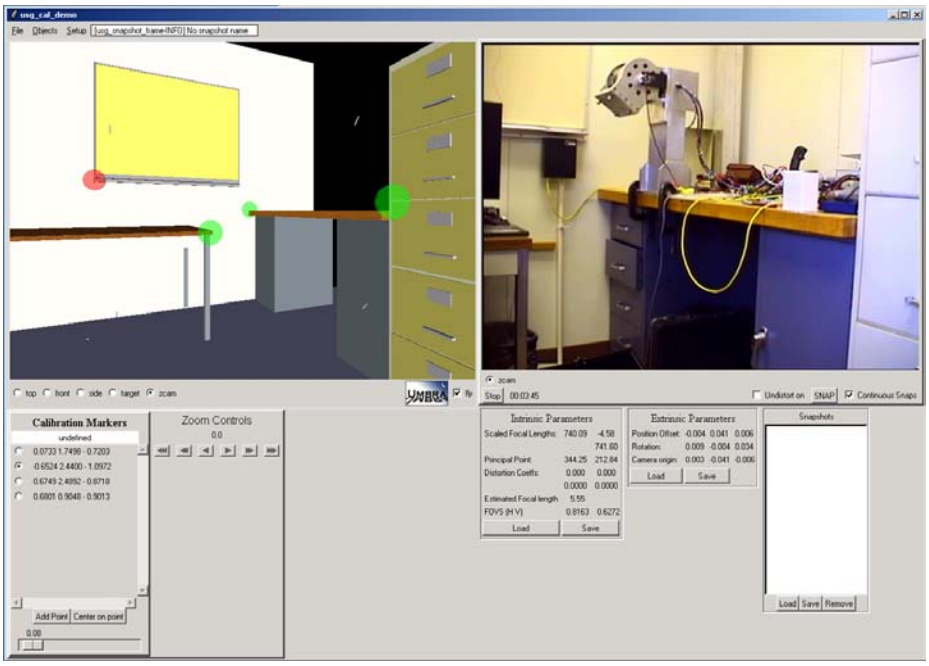


Figure 21 Preliminary Pose Estimation Tool

5.6 Digital Zoom Camera Calibration

Compact calibrated digital zoom cameras, such as the SONY-FCB-10A (Fig. 22), have just recently become available commercially. These cameras solve a critical problem inherent in existing fixed field-of-view cameras system, namely, they can scale to the task at hand. Previous visual targeting systems utilized for IED operations could only perform tasks on objects with features ranging from 4" to 18" in length. Analog zoom cameras could not be used because they were either too bulky, or because the effective field-of-view could not be accurately measured or con-

trolled. The introduction of these new compact cameras, with repeatable, digitally assigned zoom controls has allowed us to reconsider approaches for digital targeting operations.

To utilize these cameras as a precision targeting cameras, we designed a camera housing that accurately and repeatably attaches the camera to a Directed Perception pan and tilt unit, creating a small fast single camera imaging system.

Figure 22 Sony FCB-10A with Customized Housing on PTU Unit

With a pair of these small PTU system mounted on a mobile manipulator as shown in Figure 23 it will be possible to substantially improve visual targeting operations for mobile manipulators. Improved speed, improved field-of-view, and reduced issues with occlusion should all result as part of this effort.

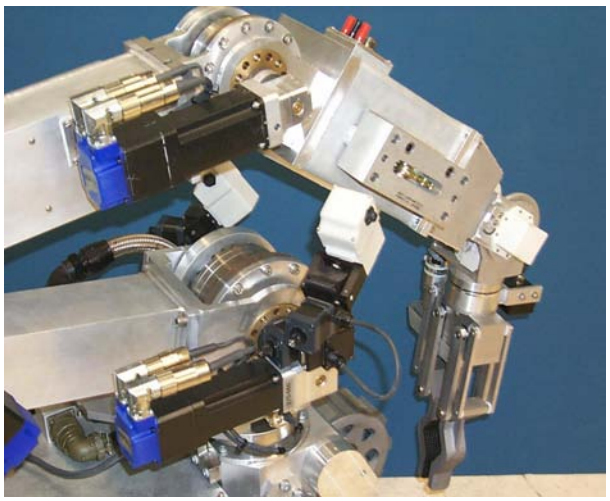


Figure 23 Mobile Manipulator With a Pair of Independent Zoom Cameras

The first step in utilizing these advanced cameras as part of a visual targeting system, is benchtop calibration. Rather than just having a static set of intrinsic and extrinsic parameters for the camera, the calibration values for a zoom camera need to be adjusted whenever the zoom value has been changed. The allowable settings for zoom are essentially continuous, with over 10000 effective settings between full wide view and fully zoom. Our approach is to compute the bench top intrinsic and extrinsic calibration values at a series of discrete settings, and use interpolation to estimate

intermediate settings. Appendix D shows the format for the XML file used for camera calibration. The zoom setting is normalized between zero and one for each particular camera, where zero represents a full wide view, and one represents full zoom.

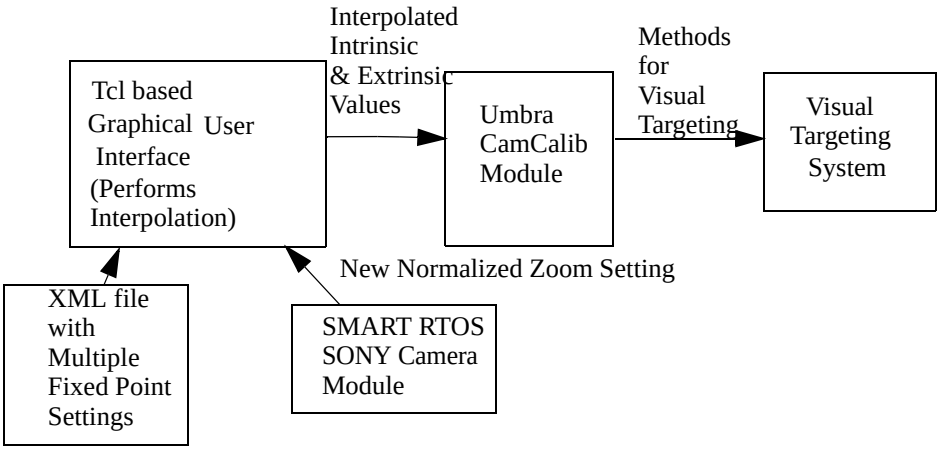


Figure 24 Zoom Camera Calibration Flowchart

The flow of parameters is shown in Figure 24 above. A SMART SONY module monitors the current state of the camera's zoom settings. These settings can either be changed automatically by the targeting system, or manually by the operator. The current zoom setting for each camera is monitored, and any change initiates a callback in the user interface. The call-back computes a new interpolation setting, and adjust the Umbra CamCalib module accordingly. Methods within the CamCalib module can then be used to perform visual targeting operations as if a fixed camera system had been used for a snapshot.

6.0 Conclusions

In this LDRD we have explored the use of visual imagery for automating operations with mobile manipulator systems used in emergency response.

During the course of this LDRD we demonstrated the use of statistical pressure snakes for automating tool pickups. The experiment consisted of a 6-degree-of-freedom (DOF) robot system with a wrist mounted camera. Special made tool pickups were designed that contained a uniquely colored a rectangular block attached to a robot tool. The robot was able to automatically approach, align itself and grab the robot tool by responding to information gleaned from the snake. Tool alignment variations in 4-DOF, (3 translation, and z-axis rotation) could be readily accommodated with this approach. The snake algorithms proved to be robust in the unstructured lighting conditions typically seen by emergency response manipulators.

We also further developed visual targeting and active sketch technologies. By providing simple object primitives to the operator, and a derived list of tasks based on the primitives, we've developed an interface which is fast, intuitive and powerful for a wide range of activities, such as grasping objects, opening doors and scanning surfaces.

We demonstrated that live video imagery could be utilized by the robotic operators within a 3-D modelling environment. Operators are often plagued with a tunnel vision phenomena, where they have the video feed, but lose the context of the video. By embedding the video within a 3-D world model, this problem is addressed. The operator is able to freely move from video perspective to overview perspective and back. The 3-D world serves as a working environment to embed new information as it is received.

Finally, several advancements in camera calibration have been made. An interactive, graphical tool for bench top calibration of intrinsic and extrinsic camera parameters has been developed. A robust algorithm for non-perspective n-point pose estimation has been developed, and is being incorporated into an interactive pose estimation tool. New zoom camera systems have been investigated, and a working continuously variable calibration model has been developed which is compatible with existing targeting systems.

7.0 References

[1] D. Perrin and C. Smith, "*Rethinking classical internal forces for active contour models*," Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2001.

[2] HP Schaub & C.E. Smith, "*Color Snakes for Dynamic Lighting Conditions on Mobile Manipulation Platforms*", Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, Oct. 2003

[3] H. Schaub, "*Statistical Pressure Snakes based on Color Images*", SAND-Report 2004-1867.

[4] H. Schaub, "*Visual Servoing Using Statistical Pressure Snakes*," SAND-Report 2004-1868.

[5] H. Schaub, "*Extracting Primary Features of a Statistical Pressure Snake*", SAND-Report 2004 1869.

[6] H. Schaub, "*Reading Color Barcodes using Visual Snakes*," SAND-Report 2004 1870.

[7] H. Schaub, *"Matching a Statistical Pressure Snake to a Four-Sided Polygon and Estimating the Polygon Corners,"* SAND 2004 1871.

[8]R. J. Anderson, "*Modular Architecture for Robotics and Teleoperation*", U.S. Patent No. 5581666. Dec. 3, 1996.

[9]R. J. Anderson, "*SMART: A Modular Control Architecture for Telerobotics*," IEEE Robotics and Automation Society Magazine, pp. 10-18, Sept. 95.

[10]R. J. Anderson, "*SMART: A Modular Control Architecture for Telerobotics*," IEEE Robotics and Automation Society Magazine, pp. 10-18, Sept. 95.

[11]E. J. Gottlieb, M. McDonald and et. all, "*The Umbra Simulation Framework as Applied to Building HLA Federates*," Proceedings of the 2002 Winter Simulation Conference, pp. 981-989

[12]E. J. Gottlieb, M. J. McDonald, and F.J. Oppel, "*The Umbra High Level Architecture Interface*," Sandia Technical Report SAND2002-0675.

[13]G., DeSouza and A. Kark, "*Vision for Mobile Robot Navigation: A Survey,*" IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24. no. 2, pp 237-267, 2002.

[14] R. W. Harrigan and P. C. Bennett, "*Making Remote Manipulators Easy to Use*", Unmanned Ground Vehicle Technology III (SPIE International) Symposium, 2001 Orlando FL.

[15] M. McDonald, R.J. Anderson and S. Gladwell, "*Active Sketch A New Human Interface*", Patent Application, filed Dec. 17, 2002. (Status pending)

[16] R. Y. Tsai, "*A Versatile Camera Calibration Technique for 3D Machine Vision*," IEEE J. Robotics & Automation, RA-3, No. 4, August 1987, pp. 323-344.

[17] J. Weng, P. Cohen and M. Herniou, "*Camera Calibration and Distortion Models and Accuracy Evaluation*," IEEE Transactions on Pattern Analysis & Machine Intelligence, vol. 14, no. 10, Oct. 1992., pp. 965-980.

[18] Z. Zhang, "*A Flexible New Technique for Camera Calibration*," IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 22, No. 11, Nov. 2000, pp. 1330-1334.

[19] http://developer.intel.com, "*Open Source Computer Vision Library: Reference Manual*," Order Number: 123456-001.

[20] C-S Chen and W.Y. Chang, "*Pose Estimation for Generalized Imaging Device via Solving Non-Perspective N Point Problem*," Proceedings of the 2002 IEEE International Conference on Robotics & Automation, Washington, DC, May 2002. pp. 2931-2937.

Appendix A: Camera Calibration Terminology

Camera calibration is the process of determining the relationship between 2-D image points and 3-D world points. There is a large body of work developing camera calibration approaches and techniques. This section defines the notation and provides an overview of the algorithms employed for computing the camera calibration (as used by the camCalib Umbra module).

A 3-D point, $\boldsymbol{x_i}$, is represented in homogenous coordinates as

$$\vec{x}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \tag{4}$$

and a 2-D image point (u_i, v_i) is represented in homogenous coordinates as

$$\vec{u}_i = \begin{bmatrix} U_i \\ V_i \\ t_i \end{bmatrix} \tag{5}$$

where

$$\begin{aligned} u_i &= \frac{U_i}{t_i} \\ v_i &= \frac{V_i}{t_i} \end{aligned} \tag{6}$$

The mapping between the two frames utilizes the camera calibration matrix.

$$C = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \end{bmatrix} = \begin{bmatrix} \vec{c}_1 \\ \vec{c}_2 \\ \vec{c}_3 \end{bmatrix} \tag{7}$$

and is given by

$$\vec{u}_i = C\vec{x}_i \tag{8}$$

It is possible to solve for the elements of *C* for a given camera by recording the *u*, *v* coordinates of various points that are defined in *x, y, z* coordinates. Following the approach in [1], but applying the notation used here

$$\begin{bmatrix} U_i \\ V_i \\ t_i \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \tag{9}$$

Expanding the inner products and utilizing equation 6 gives the following mapping (where c_{34} has been set to one since the scaling on the matrix is arbitrary).

$$\begin{bmatrix} x_i & y_i & z_i & 1 & 0 & 0 & 0 & 0 & -u_i x_i & -u_i y_i & -u_i z_i \\ 0 & 0 & 0 & 0 & x_i & y_i & z_i & 1 & -v_i x_i & -v_i y_i & -v_i z_i \end{bmatrix} \begin{bmatrix} c_{11} \\ c_{12} \\ c_{13} \\ c_{14} \\ c_{21} \\ c_{22} \\ c_{23} \\ c_{24} \\ c_{31} \\ c_{32} \\ c_{33} \end{bmatrix} = \begin{bmatrix} u_i \\ v_i \end{bmatrix} \tag{10}$$

This represents a linear equation in the unknown parameters c_{ii} which is valid for pair of *xyz* and *uv* parameters. By applying this equation to *n* sample pairs (x_1 through x_n) a overdetermined system of equations can result, as shown below:

$$\begin{bmatrix} x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & -u_1 x_1 & -u_1 y_1 & -u_1 z_1 \\ 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 & -v_1 x_1 & -v_1 y_1 & -v_1 z_1 \\ & & & & & . & & & & & \\ x_n & y_n & z_n & 1 & 0 & 0 & 0 & 0 & -u_n x_n & -u_n y_n & -u_n z_n \\ 0 & 0 & 0 & 0 & x_n & y_n & z_n & 1 & -v_n x_n & -v_n y_n & -v_n z_n \end{bmatrix} \begin{bmatrix} c_{11} \\ c_{12} \\ c_{13} \\ c_{14} \\ c_{21} \\ c_{22} \\ c_{23} \\ c_{24} \\ c_{31} \\ c_{32} \\ c_{33} \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ . \\ u_n \\ v_n \end{bmatrix} \tag{11}$$

This is in the form of a the equation $Ax = b$ which can be solved in a least squares optimal sense by using the pseudo-inverse of A.

$$x = A^t b = A^T(AA^T)^{-1}b \qquad (12)$$

This gives both a representation and an algorithm for computing the full camera calibration matrix in terms of 11 free parameters. By taking at least 6 non-planar (i.e., all points should not exist in the same plane) points the rank of A should be greater than 11 and a valid solution will be found. If desired, The parameters in C can be further optimized by performing a non-linear optimization (e.g., Levinburgh-Marquadt) to minimize the square of the error in

$$\min \sum_i \left(\left\| \begin{bmatrix} u_i \\ v_i \end{bmatrix} - C \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \right\|^2 \right) \qquad (13)$$

Appendix B: Using the Extrinsic/Intrinsic Reference Model

The algorithm provided a representation that utilizes 11 parameters, but it does little to separate what are considered to be the intrinsic parameters of the camera (i.e., focal lengths and distortion) and the extrinsic parameters (i.e., the transformation matrix that locates the camera in a workcell). Thus if any one parameter were to change, this model is unsuited to track the changes and would require a recomputing of the complete camera matrix.

Consider the mapping below which breaks out the camera calibration matrix into two sub-matrices, the intrinsic calibration matrix C_{int}, and the extrinsic calibration matrix, C_{ext}

$$C = C_{int}C_{ext} = \begin{bmatrix} f_x & f_{xy} & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_y \\ r_{31} & r_{32} & r_{33} & d_z \end{bmatrix} = \begin{bmatrix} f_x & f_{xy} & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R & d \end{bmatrix} \qquad (14)$$

where f_x is the x-axis focal length, f_y is the y-axis focal length, u_0 and v_0 are the image plane offsets, R, is a orthonormal rotation matrix and d is the translation. The parameter f_{xy} represents a skew term that is ideally zero, but is required in order for there to be a one-to-one mapping between parameter representations

This would appear to have seventeen independent variables, but because the matrix R is required to be orthonormal, there are 6 additional constraints on the form of R, and thus only eleven independent parameters. It is possible to solve for these parameters in terms of the camera calibration matrix elements. This is shown in the approach below.

Define the 3x1 vectors c1, c2, c3 as shown below

$$C = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \end{bmatrix} = \begin{bmatrix} \vec{c}_1 & c_{14} \\ \vec{c}_2 & c_{24} \\ \vec{c}_3 & c_{34} \end{bmatrix} \qquad (15)$$

and define the rotation matrix as a series of row vectors

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} \vec{r}_1 \\ \vec{r}_2 \\ \vec{r}_3 \end{bmatrix} \qquad (16)$$

then the resulting equation is obtained

$$\bar{C} = \begin{bmatrix} \vec{c}_1 & c_{14} \\ \vec{c}_2 & c_{24} \\ \vec{c}_3 & c_{34} \end{bmatrix} = \begin{bmatrix} f_x & f_{xy} & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & d_x \\ r_{21} & r_{22} & r_{23} & d_y \\ r_{31} & r_{32} & r_{33} & d_z \end{bmatrix} = \begin{bmatrix} f_x & f_{xy} & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \vec{r}_1 & d_x \\ \vec{r}_2 & d_y \\ \vec{r}_3 & d_z \end{bmatrix} \quad (17)$$

From the bottom row of this equation the following equalities:

$$\begin{aligned} d_z &= c_{34} \\ \vec{c}_3 &= \vec{r}_3 \end{aligned} \quad (18)$$

Because the matrix C can be arbitrarily scaled and the vector r3 must have a unit norm, it is clear the matrix $\bar{C}$ must first be normalized accordingly.

$$\bar{C} = \frac{C}{\sqrt{c_{31}^2 + c_{32}^2 + c_{33}^2}} \quad (19)$$

In this case the term c34 is no longer unit valued, but directly gives the value for d_z.

From the second row of Eq. 17,

$$\vec{c}_2 = f_y \vec{r}_2 + v_0 \vec{r}_3 \quad (20)$$

Multiplying both sides by r3 and taking advantage of the orthonormal requirement on R gives

$$\vec{r}_3^{\,T} \vec{c}_2 = \vec{r}_3^{\,T} (f_y \vec{r}_2 + v_0 \vec{r}_3) = v_0 \quad (21)$$

Once v_0 is known then the focal length f_y and rotation vector r3 can be retrieved by utilizing the unit value condition on r3.

$$f_y = \left\| \vec{c}_2 - v_0 \vec{r}_3 \right\| \quad (22)$$

$$\vec{r}_2 = \frac{\vec{c}_2 - v_0 \vec{r}_3}{f_y} \quad (23)$$

The top row of Eq. 17 gives us

$$\vec{c}_1 = f_x \vec{r}_1 + f_{xy} \vec{r}_2 + u_0 \vec{r}_3 \quad (24)$$

Using the same orthonormal constraints on R gives us

$$u_0 = \vec{r}_3^{\,T} \vec{c}_1 \quad (25)$$

$$f_{xy} = \vec{r}_2^{\,T} \vec{c}_1 \quad (26)$$

$$f_x = \left\| \vec{c_1} - f_{xy}\vec{r_2} - u_0\vec{r_3} \right\| \quad (27)$$

$$\vec{r_1} = \frac{\vec{c_1} - f_{xy}\vec{r_2} - u_0\vec{r_3}}{f_x} \quad (28)$$

Finally, by computing the second column variables the values for d_y and d_x are derived.

$$d_y = \frac{c_{24} - v_0 d_z}{f_y} \quad (29)$$

$$d_x = \frac{c_{14} - u_0 d_z - f_{xy} d_y}{f_x} \quad (30)$$

This set of equations provides the mappings to create the intrinsic and extrinsic parameter files from the camera calibration matrix given an undistorted image.

Appendix C: Solving the non-Perspective n-Point problem

For a series of XYZ points attached to an object frame, viewed from a imaging device (i.e., PTU camera), the imaging equation relating the image coordinates to XYZ points is given by

$$\begin{bmatrix} U_i \\ V_i \\ t_i \end{bmatrix} = C_{int} \begin{bmatrix} & 0 & \\ I_3 & 0 & \\ & 0 & \end{bmatrix} T_{ext} T_{kin} T_{dev} T_{obj}^{-1} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \qquad (31)$$

Typically T_{kin} may vary for each point taken but is well known, T_{ext} is known and fixed, C_{int} is considered known (but may vary based on zoom settings), and the goal of the pose estimation is to determine either T_{dev} or T_{obj} accurately, assuming every other transform is known for the given view, and that the value for either of these transforms is fixed for multiple measurements.

Without loss of generality equation 31 simplifies to

$$\begin{bmatrix} U_i \\ V_i \\ t_i \end{bmatrix} = C_{int} \begin{bmatrix} & 0 & \\ I_3 & 0 & \\ & 0 & \end{bmatrix} T_A T_{est} \begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ \hat{z}_i \\ 1 \end{bmatrix} \qquad (32)$$

where depending on which transforms are known a priori.

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ \hat{z}_i \\ 1 \end{bmatrix} = T_{obj}^{-1} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \qquad T_A = T_{ext} T_{kin} \qquad T_{est} = T_{dev} \text{ or} \qquad (33)$$

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ \hat{z}_i \\ 1 \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \qquad T_A = T_{ext} T_{kin} T_{dev} \qquad T_{est} = T_{obj}^{-1} \qquad (34)$$

Defining the transform

$$T_A = \begin{bmatrix} R_A & d_A \\ 0_{3/1} & 1 \end{bmatrix} \qquad (35)$$

it follows that

$$\begin{bmatrix} U_i \\ V_i \\ t_i \end{bmatrix} = C_{int} \begin{bmatrix} 0 \\ I_3 \;\; 0 \\ 0 \end{bmatrix} T_A T_{est} \begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ \hat{z}_i \\ 1 \end{bmatrix} = C_{int} \begin{bmatrix} R_A & d_A \end{bmatrix} T_{est} \begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ \hat{z}_i \\ 1 \end{bmatrix} \qquad (36)$$

Representing the estimated transform using a first order approximation

$$T_{est} = \begin{bmatrix} R_{est} & d_{est} \\ 0_{3/1} & 1 \end{bmatrix} = \begin{bmatrix} R_0 + X(\Delta\theta)R_0 & d_0 + \Delta d \\ 0_{3/1} & 1 \end{bmatrix} \qquad (37)$$

where $X(\;\;)$ is the skew-symmetry operator, $R_0 = e^{X(\theta_0)}$ is the initial rotation estimate, d_0 is the initial distance estimate, $\Delta\theta$ is the angular correction, and Δd is the position correction.

$$\begin{aligned} \begin{bmatrix} U_i \\ V_i \\ t_i \end{bmatrix} &= C_{int} \begin{bmatrix} R_A & d_A \end{bmatrix} \begin{bmatrix} R_0 + X(\Delta\theta)R_0 & d_0 + \Delta d \\ 0_{3/1} & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ \hat{z}_i \\ 1 \end{bmatrix} \\ &= C_{int}R_AR_0 \begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ \hat{z}_i \end{bmatrix} + C_{int}R_AX(\Delta\theta)R_0 \begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ \hat{z}_i \end{bmatrix} + C_{int}R_Ad_0 + C_{int}d_A + C_{int}R_A\Delta d \qquad (38) \\ &= C_{int}R_AR_0 \begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ \hat{z}_i \end{bmatrix} + C_{int}R_Ad_0 + C_{int}d_A - C_{int}R_AX\left(R_0 \begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ \hat{z}_i \end{bmatrix} \right) \Delta\theta + C_{int}R_A\Delta d \end{aligned}$$

substituting again gives

$$\begin{bmatrix} u_it_i \\ v_it_i \\ t_i \end{bmatrix} = \begin{bmatrix} \lambda_{10} \\ \lambda_{20} \\ \lambda_{30} \end{bmatrix} + \begin{bmatrix} \lambda_{11} & \lambda_{12} & \lambda_{13} & \lambda_{14} & \lambda_{15} & \lambda_{16} \\ \lambda_{21} & \lambda_{22} & \lambda_{23} & \lambda_{24} & \lambda_{25} & \lambda_{26} \\ \lambda_{31} & \lambda_{32} & \lambda_{33} & \lambda_{34} & \lambda_{35} & \lambda_{36} \end{bmatrix} \begin{bmatrix} \Delta\theta \\ \Delta d \end{bmatrix} = \begin{bmatrix} \lambda_{10} \\ \lambda_{20} \\ \lambda_{30} \end{bmatrix} + \begin{bmatrix} \vec{\lambda}_1 \\ \vec{\lambda}_2 \\ \vec{\lambda}_3 \end{bmatrix} \Phi \qquad (39)$$

where

$$\begin{bmatrix}\lambda_{10}\\ \lambda_{20}\\ \lambda_{30}\end{bmatrix} = C_{int}R_A R_0 \begin{bmatrix}\hat{x}_i\\ \hat{y}_i\\ \hat{z}_i\end{bmatrix} + C_{int}R_A d_0 + C_{int}d_A$$

$$= C_{int}\left(R_A\left(R_0 \begin{bmatrix}\hat{x}_i\\ \hat{y}_i\\ \hat{z}_i\end{bmatrix} + d_0\right) + d_A\right) \qquad (40)$$

$$\begin{bmatrix}\lambda_{11} & \lambda_{12} & \lambda_{13}\\ \lambda_{21} & \lambda_{22} & \lambda_{23}\\ \lambda_{31} & \lambda_{32} & \lambda_{33}\end{bmatrix} = -C_{int}R_A X\left(R_0 \begin{bmatrix}\hat{x}_i\\ \hat{y}_i\\ \hat{z}_i\end{bmatrix}\right) \qquad (41)$$

$$\begin{bmatrix}\lambda_{14} & \lambda_{15} & \lambda_{16}\\ \lambda_{24} & \lambda_{25} & \lambda_{26}\\ \lambda_{34} & \lambda_{35} & \lambda_{36}\end{bmatrix} = C_{int}R_A \qquad (42)$$

Finally, solving for Φ and substituting

$$t_i = \lambda_{30} + \overrightarrow{\lambda_3}\Phi \qquad (43)$$

$$\begin{bmatrix}u_i(\lambda_{30} + \overrightarrow{\lambda_3}\Phi)\\ v_i(\lambda_{30} + \overrightarrow{\lambda_3}\Phi)\end{bmatrix} = \begin{bmatrix}\lambda_{10}\\ \lambda_{20}\end{bmatrix} + \begin{bmatrix}\overrightarrow{\lambda_1}\\ \overrightarrow{\lambda_2}\end{bmatrix}\Phi \qquad (44)$$

$$\begin{bmatrix}\lambda_{30}u_i - \lambda_{10}\\ \lambda_{30}v_i - \lambda_{20}\end{bmatrix} = \begin{bmatrix}\overrightarrow{\lambda_1} - u_i\overrightarrow{\lambda_3}\\ \overrightarrow{\lambda_2} - v_i\overrightarrow{\lambda_3}\end{bmatrix}\Phi \qquad (45)$$

This is of the linear equation form $b_i = A_i\Phi$

A least squares fit is then obtained by taking multiple points and fitting the data.

$$\begin{bmatrix}b_1\\ b_2\\ \cdots\\ b_n\end{bmatrix} = \begin{bmatrix}A_1\\ A_2\\ \cdots\\ A_n\end{bmatrix}\Phi = b = A\Phi \qquad (46)$$

$$\Phi = A^{\perp}b = (A^T A)^{-1}A^T b \qquad (47)$$

In summary, the algorithm is as follows.

First, compute the least squares equation 47 to solve for $\Phi = \begin{bmatrix} \Delta\theta \\ \Delta d \end{bmatrix}$ where

$$A_i = \begin{bmatrix} \vec{\lambda_1} - u_i\vec{\lambda_3} \\ \vec{\lambda_2} - v_i\vec{\lambda_3} \end{bmatrix}, b_i = \begin{bmatrix} \lambda_{03}u_i - \lambda_{01} \\ \lambda_{03}v_i - \lambda_{02} \end{bmatrix} \quad (48)$$

Second, compute a new rotation *R* matrix

$$\begin{aligned} \theta_0(k+1) &= \theta_0(k) + \Delta\theta \\ R_0(k+1) &= e^{X(\theta_0 + \Delta\theta)} \\ d_0(k+1) &= d_0(k) + \Delta d \end{aligned} \quad (49)$$

Using the axis angle representation,

$$\begin{aligned} \phi &= \|\theta_0\| \\ \hat{k} &= \frac{\theta_0}{\|\theta_0\|} \end{aligned} \quad (50)$$

the new rotation matrix can be written as

$$R = \cos(\phi)I_3 + (1 - \cos(\phi))\hat{k}\hat{k}^T + \sin(\phi)X(\hat{k}) \quad (51)$$

Appendix D: XML Format for Zoom Camera Calibration

The XML format defines a series of intrinsic and extrinsic camera settings associated with a zoom camera. The Zoom camera is assumed to range between a normalized zoom range of 0.0 for full wide view, and 1.0 for a fully zoomed view. The XML file uses the following keywords:

SR_cam: Defines the name of the camera object, and encompasses each recorded normalized zoom setting.

SR_zoom: Contains the normalized zoom setting at which any intrinsic and extrinsic settings are valid.

SR_cam_int: Specifies the intrinsic parameters, and should be followed by the seven intrinsic parameter values: $\left\{ f_x\ f_y\ f_{xy}\ u_0\ v_0\ k_1\ k_2\ p_1\ p_2 \right\}$

SR_cam_ext: Specifies the extrinsic parameters, and should be followed by the six extrinsic parameter values: $\left\{ d_x\ d_y\ d_z\ \theta_x\ \theta_y\ \theta_z \right\}$

The XML file for a calibrated SONY zoom camera is shown below:

```
<?xml version="1.0"?>
<SR_world>
<SR_filename> C:/ieddApps/camera/zoom_cam.xml </SR_filename>
<SR_comment>
Sony Zoom Camera Values  Used for calibration of  Sony FCB camera.
</SR_comment>

<SR_cam> zcam
  <SR_zoom> 0.0
  <SR_cam_int>{740.09 741.60 -4.58 344.25 212.84 0. 0. 0. 0.} </SR_cam_int>
    <SR_cam_ext> {-0.004 0.041 0.006 0.009 -0.004 0.034   } </SR_cam_ext>
  </SR_zoom>

  <SR_zoom> 0.025
    <SR_cam_int> {794.40 800.52 -6.28 327.35 251.45 0.0 0. 0. 0.} </SR_cam_int>
    <SR_cam_ext> {-0.004 0.040 0.009 0.056 0.016 0.032   } </SR_cam_ext>
  </SR_zoom>

  <SR_zoom> 0.05
    <SR_cam_int> {844.86 849.04 -7.24 316.87 219.260 0. 0. 0. 0. } </SR_cam_int>
    <SR_cam_ext> {-0.002 0.041 -0.000 0.022 0.022 0.032  } </SR_cam_ext>
 </SR_zoom>

  <SR_zoom> 0.1
    <SR_cam_int> {1004.12 1005.49 -5.17 329.62 227.659 0. 0. 0. 0.} </SR_cam_int>
    <SR_cam_ext> {-0.002 0.041 0.008 0.034 0.004 0.033    } </SR_cam_ext>
  </SR_zoom>

  <SR_zoom> 0.15
    <SR_cam_int> {1208.51 1207.932 -5.85 358.46 249.57 0. 0. 0. 0.} </SR_cam_int>
   <SR_cam_ext> {-0.002 0.041 0.014 0.052 -0.020 0.033    } </SR_cam_ext>
  </SR_zoom>

  <SR_zoom> 0.3
```

```
    <SR_cam_int> {2314.0 2319.32 -15.74 321.77 139.95 0. 0. 0. 0.} </SR_cam_int>
   <SR_cam_ext> {-0.002 0.042 0.019 0.001 0.003 0.034  }   </SR_cam_ext>
  </SR_zoom>

  <SR_zoom> 0.4
    <SR_cam_int> {3500.0 3500.0 0.0 320.0 240.0 0.0 0.0 0.0 0.0} </SR_cam_int>
    <SR_cam_ext> {-0.001 0.041 0.021 0.078 0.051 0.032  } </SR_cam_ext>
  </SR_zoom>

  <SR_zoom> 1.0
    <SR_cam_int> {29600.0 29600.0 0.0 320.0 240.0 0.0 0.0 0.0 0.0} </SR_cam_int>
    <SR_cam_ext> {-0.001 0.041 0.021 0.022 0.022 0.032     } </SR_cam_ext>
   </SR_zoom>
</SR_cam>
</SR_world>
```

8.0 Distribution:

10	MS1125	Robert J. Anderson	15244
2	MS1125	Phil Bennett	15244
1	MS1003	John Feddema	15211
1	MS9018	Central Technical File	8945-1
2	MS0899	Technical Library	9616
1	MS0123	LDRD Office (Donna Chavez)	1011